International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 01 | Jan - 2025 SJIF Rating: 8.448 ISSN: 2582-3930

Generating Music Using Machine Learning

Vishesh R Bhaddurgatte[1], Shushruth S[2]
[1]*Electronics and Communication Engineering ,PES University*
[2] *Electronics and Communication Engineering ,PES University*

Abstract – ***This report delves into the realm of music generation using advanced machine learning techniques, focusing on Recurrent Neural Networks (RNNs), Generative Adversarial Networks (GANs), and their amalgamation in the form of RNN-GAN. The objective is to explore the capabilities of these models in creating novel and expressive musical compositions.***

The examines the fundamental concepts of RNNs, known for their prowess in sequence modelling, making them ideal for capturing the temporal intricacies present in music. Additionally, it explores the generative potential of GANs, that employs adversarial training to produce realistic and innovative musical sequences.

The synthesis of RNNs and GANs into an integrated RNN-GAN framework is investigated. This hybrid approach aims to leverage the strengths of both architectures, enabling the generation of coherent and diverse musical pieces.

I.INTRODUCTION

In the fascinating world where technology meets tunes, this report explores how computers can create music. Imagine it as a collaboration between machines and melodies, where we use smart algorithms to generate musical compositions. Our journey revolves around three main things: Recurrent Neural Networks (RNNs), Generative Adversarial Networks (GANs), and a mix of both, known as RNN-GANs.

Music has always been a powerful way for people to express feelings and tell stories. Now, with the rise of artificial intelligence, we are curious about how machines can contribute to this creative process. We start by checking out RNNs, special algorithms designed to understand patterns and structures in sequences, like the ones found in music.

Then, we dive into GANs, which bring a whole new twist to creating music. The exciting part comes when we blend RNNs and GANs together into the RNN-GAN model, creating a mix that is both good at understanding patterns and great at making new music. As we go forward, we dream of a future where AI does not just copy what humans do but adds something new to our creative mix. We are thinking about attention-grabbing tricks, teaming up with musicians, making music on the spot, and always being careful about how we use this tech.

In the evolving landscape of artificial intelligence and machine learning, the intersection with creative domains has sparked profound interest and innovation. One such captivating realm is the generation of music through computational models

This report embarks on an exploration of advanced machine learning techniques, with a specific focus on Recurrent Neural Networks, Generative Adversarial Networks, and the hybrid model, RNN-GAN, to unravel the intricate process of creating music through artificial intelligence.

Generating music using machine learning involves training a computer program to learn patterns and structures from existing musical data, and then using that knowledge to create new music.

To train a neural network to generate music, researchers typically feed it with large amounts of musical data, such as MIDI files or sheet music.

The network then learns to recognize patterns in this data, such as common chord progressions, melodies, and rhythms, and uses this knowledge to generate new music that follows similar patterns.

The idea was to develop a machine learning model that can generate music autonomously, given a set of musical inputs as training data and to create a tool that can aid musicians and composers in their creative process, by providing them with a source of inspiration and new ideas.

II. METHODOLOGY

1) PROPOSED METHODOLOGY: We propose a recurrent neural network architecture, C-RNN-GAN (Continuous RNN-GAN), that is trained with adversarial training to model the whole joint probability of a sequence, and to be able to generate sequences of data. We have trained our model on sequences of classical music. The proposed model is a recurrent neural network with adversarial training. The adversaries are two different deep recurrent neural models, a generator (G) and a discriminator (D). The generator is trained to generate data that is indistinguishable from real data, while the discriminator is trained to identify the generated data. The training becomes a zero-sum game when the generator produces data that the discriminator cannot tell from real data. The input to each cell in G is a random vector, concatenated with the output of previous cell. Feeding the output from the previous cell is common practice when training RNNs as language models, and has also been used in music composition. The discriminator consists of a bidirectional recurrent net, allowing it to take context in both directions into account for its decisions. In this work, the recurrent network used is the Long short-term memory. It has an internal structure with gates that help with the vanishing gradient problem, and to learn longer dependencies.

Model Layout Details: Both the Generator (G) and Discriminator (D) utilize LSTM networks with a depth of 2, featuring 350 internal units per LSTM cell. Notably, D employs a bidirectional layout, while G is unidirectional.
The output from each LSTM cell in D feeds into a fully connected layer with shared weights across time steps. A sigmoid output per cell is averaged to determine the final decision for the sequence.

Baseline: The baseline comprises a recurrent network similar to the generator, trained solely to predict the next tone event at each recurrence point.

Dataset: Training data, sourced from the web as MIDI files of classical music, includes well-known works. Each MIDI event of the type "note on" is recorded with its duration, tone, intensity, and time since the last tone. The tone data is internally represented with the corresponding sound frequency. All data is normalized to a tick resolution of 384 per quarter note. The dataset comprises 3697 MIDI files from 160 different classical music composers.

Training: Backpropagation through time (BPTT) and mini-batch stochastic gradient descent are employed. A learning rate of 0.1 is set, and L2 regularization is applied to the weights in both G and D. The model undergoes pretraining for six epochs with a squared error loss for predicting the next event. During pretraining, a random vector v, concatenated with the output at the previous time step, serves as the input to each LSTM cell. Adversarial training follows, where freezing is applied to D and G when their training losses deviate by a significant margin. Feature matching is also utilized to encourage greater variance in G, replacing the standard generator loss with the objective to produce an internal representation that matches real data in the discriminator.

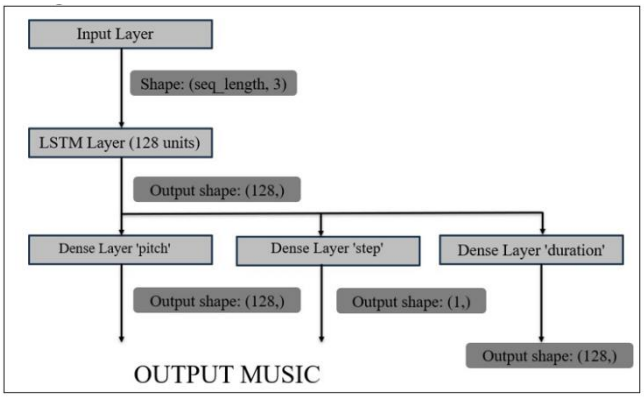

Fig -1:RNN Model Block Diagram

2) OTHER METHODS IMPLEMENTED:

A) RECURRENT NEURAL NETWORK MODEL: Recurrent Neural Networks (RNNs) are employed in music generation to capture temporal dependencies and patterns in musical sequences. Musical pieces are represented as sequences of encoded notes or chords, with input encoding using methods like one-hot encoding or embedding layers. Architectures like Long Short-Term Memory (LSTM) networks are preferred for their ability to handle long-term dependencies. During training, the model learns to predict the next element in a sequence, adjusting parameters to minimize prediction errors. Once trained, the RNN generates new music by providing an initial seed sequence. Post-processing steps and evaluation metrics ensure adherence to musical rules and assess the quality of the generated compositions. Our dataset contains 1200+ paired audio and MIDI recordings from ten years of International Piano-e- Competition. These MIDI files will be used for training the music generation model.

MIDI File Processing: We select a sample MIDI file for processing and extract notes from it. This is done by analyzing the instrument tracks within the MIDI file and recording information about each note, including its pitch, start time, end time, step, and duration.

Training Dataset Creation: The extracted notes from multiple MIDI files are concatenated to create a larger dataset. This dataset is then processed to create sequences of notes, which will be used for training the music generation model. The model architecture includes an LSTM layer followed by separate output. layers for pitch, step, and duration prediction. After training, the model is used to generate new notes. It takes the sequence of notes and predicts the next note's pitch, step, and duration. This process is repeated to generate new music.

Model Details:
Input Layer: The input layer takes sequences of shape (seq_length, 3) as input. Each sequence contains information about pitch, step (time step), and duration.

LSTM Layer: This LSTM layer processes the input sequences and has 128 units (or cells). It captures temporal dependencies and patterns in the input data.

Dense Layers:
Dense Layer 'pitch': This dense (fully connected) layer follows the LSTM layer and has 128 units. It predicts the pitch of the next note in the music sequence.
Dense Layer 'step': Another dense layer with a single unit follows the LSTM layer. It predicts the time step until the next note.
Dense Layer 'duration': This dense layer also has a single unit and predicts the duration of the next note.
Outputs: There are three output branches from the model, each corresponding to one of the predicted attributes: 'pitch', 'step', and 'duration'. The output shapes are specified for each branch.

B) GENERATIVE ADVERSIAL NETWORK MODEL:
Generative Adversarial Networks (GANs) are employed in music generation to produce realistic and diverse musical sequences. Consisting of a generator and a discriminator, GANs engage in an adversarial training process. The generator creates music sequences from random latent vectors, while the discriminator evaluates their authenticity. Through this competitive interplay, the generator refines its ability to produce convincing music, and the discriminator improves its discernment between real and generated sequences. GANs

operate in a latent space, allowing for the exploration of diverse musical styles. Post-processing steps can be applied to control the generated music's style, and evaluation often involves subjective assessments of musical quality and coherence. GANs offer a dynamic and innovative approach to music generation, pushing the boundaries of creativity and enabling the generation of novel musical compositions.MIDI File Processing: We select a sample MIDI file for processing and extract notes from it. This is done by analyzing the instrument tracks within the MIDI file and recording information about each note, including its pitch, start time, end time, step, and duration. The dataset was converted into a NumPy array to correctly parse each sample into its corresponding piano roll format which was then used for training the GAN system. Generative Adversarial Networks (GANs) are employed, consisting of a generator and discriminator. A generator transforms random noise into synthetic samples, while the discriminator estimates the authenticity of samples. Training continues iteratively until convergence, ensuring the model cannot differentiate between real and synthetic samples. Parameters such as batch size, shuffle buffer size, input size, and latent noise vector size are carefully chosen for stability and variability in generated music.

Model Details: GANs are unsupervised neural network models that use two competing supervised sub-models to generate deep representations of the input data without the explicit need for annotations.
The two sub-models work in concert to develop high-dimensional distribution of the input data that generalises well on unseen samples. The adversarial architecture behind a GAN implementation, are the two main components, the generator and the discriminator:
The generator transforms a random noise vector into a synthetic sample, which resembles real samples drawn from a distribution of the real content and the discriminator estimates the probability that a sample came from the real data rather than from the generator.

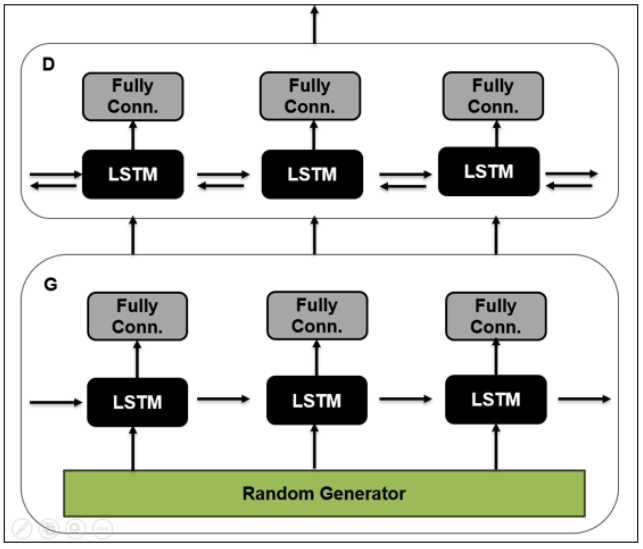

Fig -2: C-RNN-GAN Block Diagram

III. IMPLEMENTATION

RNN Implementation:
Data Collection and Preprocessing:
Obtained the MIDI dataset, specifically the Maestro dataset, which contains a diverse collection of classical piano music in MIDI format. Conducted data preprocessing to ensure uniformity and quality. Removed any corrupted or incomplete MIDI files.
Discovered key insights into the dataset's characteristics, such as the distribution of musical notes, common pitch ranges, and variations in note durations.

Model Development:
Created a neural network architecture for music generation. The architecture consists of LSTM layers for sequence modelling and dense layers for generating pitch, step, and duration. We chose the Adam optimizer and selected loss functions, including Sparse Categorical Cross-Entropy for pitch prediction.

Training:
Conducted multiple training epochs to allow the model to learn complex musical patterns. Implemented batch processing to efficiently train the model. Batches of MIDI sequences were processed in parallel, optimizing both memory usage and training speed.
Defined a sequence length that determines the length of input sequences for training

Model Evaluation:
Evaluated the model's performance using various metrics, including loss values. For further analysis we used data visualization techniques for graphical analysis of piano rolls, pitch, step, and duration of the output for comparison with the input.

GAN Implementation:
The dataset was converted into a NumPy array to correctly parse each sample into its corresponding piano roll format which was then used for training the GAN system.
This input is passed through the GAN for training, where the following happens:
The generator G, transforms a random noise vector into a synthetic sample, which resembles real samples drawn from a distribution of the real content. The discriminator D, estimates the probability that a sample came from the real data rather than from the generator G. One sample was generated from the final test of the 1000 iterations and if a person were to listen to the music snippets, noticeable variations in the tones and pitches can be observed thereby validating the experimental methodology. The music generated using GAN yielded slightly better results when compared to the previous RNN model.

C-RNN-GAN Implementation:
Model layout details: The LSTM network in both G and D has depth 2, each LSTM cell has 350 hidden units. D has a bidirectional layout, while G is unidirectional. The output from each LSTM cell in D are fed into a fully connected layer with weights shared across time steps, and one sigmoid output per cell is then averaged to the final decision for the sequence.

Dataset: We have used piano music data. The data contains

MODEL	MSE	RMSE
RNN	0.255	0.504
GAN	0.622	0.788
C-RNN-GAN	0.040	0.2

3697 midi files from 160 different composers of classical music.

Training: Learning rate was set to 0.1, and we apply L2 regularization to the weights both in G and D. The model

Table -1:Model Comparison

was pretrained for 6 epochs. Just as in the adversarial setting, the input to each LSTM cell is a random vector v, concatenated with the output at previous time step. During pretraining, we used a schema for sequence length, beginning with short sequences, randomly sampled from the training data, eventually training the model with increasingly long sequences. During adversarial training, we noticed that D can become too strong, resulting in a gradient that cannot be used to improve G. This effect is particularly clear when the network is initialized without pretraining. For this reason, we apply freezing which means stopping the updates of D whenever its training loss is less than 70% of the training loss of G. Normally, the objective for the generator is to maximize the error that the discriminator makes ,but with feature matching, the objective is instead to produce an internal representation at some level in the discriminator that matches that of real data.

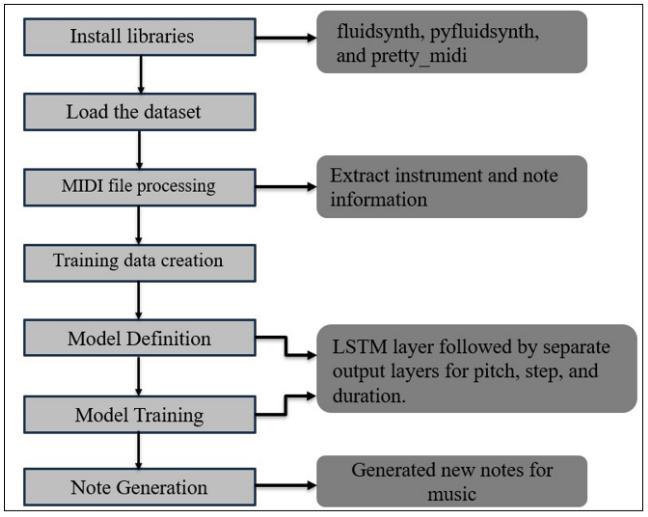

Fig -3:Implementation Workflow

IV. RESULT

MSE measures the average squared difference in audio samples between the generated and reference music.
RMSE provides a more interpretable measure of the average difference in audio samples, and it is in the same units as the audio data.
In summary, MSE and RMSE are used to quantify how well predictions or model outputs match observed values or ground truth data.
$MSE = (1/n) * \Sigma(yi - \hat{y}i)^2$
• n is the number of data points.
• yi represents the actual value.
• $\hat{y}i$ represents the generated.
$RMSE = \sqrt{(MSE)}$

V. CONCLUSION

This study has delved into the exciting intersection of artificial intelligence and music generation, leveraging the capabilities RNNs, GANs, and their hybrid integration in the form of RNN-GAN. The exploration of these models has provided valuable insights into their individual strengths and the synergies they create when combined. The findings and conclusions are:
• Sequential Modelling with RNNs: The use of RNNs, particularly Long Short-Term Memory (LSTM) networks, has demonstrated their efficacy in capturing temporal dependencies in musical sequences. The trained RNN serves as a valuable component in guiding the generative process, infusing coherence, and structure into the compositions.
• Generative Power of GANs: GANs, through adversarial training, showcase their ability to generate realistic and diverse musical sequences. The adversarial interplay between the generator and discriminator results in compositions that emulate the characteristics of the training data, introducing a novel paradigm for creative music generation.
• Hybrid Approach with RNN-GAN: The integration of RNNs and GANs into an RNN-GAN framework offers a holistic approach, combining sequential understanding with generative capabilities. The RNN's influence on the GAN's generative process enhances the quality and coherence of the generated music, showcasing the potential of this hybrid model.

VI. ACKNOWLEDGMENT

We would like to express our sincere gratitude to Electronics and Communication Department for their invaluable guidance and support throughout this research. Their insights and expertise greatly contributed to the quality and depth of this work.

We are also thankful to PES University for providing the necessary resources and facilities that enabled us to conduct this study effectively.

Special thanks go to our colleagues and peers for their constructive feedback and encouragement, which helped refine our ideas and methodologies. Finally, we extend our deepest appreciation to our families and friends for their unwavering support and understanding during the course of this research

VII. REFRENCES

1. Yang, Li-Chia, Szu-Yu Chou, and Yi-Hsuan Yang. "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation." arXiv preprint arXiv:1703.10847 (2017).

2. Arora, Sofia, et al. "An Analysis of Implementing a GAN to Generate MIDI Music." 2022 IEEE MIT Undergraduate Research Technology Conference (URTC). IEEE, 2022.

3. Wu, Jian, et al. "A hierarchical recurrent neural network for symbolic melody generation." IEEE transactions on cybernetics 50.6 (2019): 2749-2757.

4. Sajad, Sahreen, S. Dharshika, and Merin Meleet. "Music Generation for Novices Using Recurrent Neural Network (RNN)." 2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES). IEEE, 2021.

5. Su, Yuping, et al. "Folk melody generation based on CNN-BiGRU and Self-Attention." 2022 4th International Conference on Communications, Information System and Computer Engineering (CISCE). IEEE, 2022.

6. Shah, F., Naik, T. and Vyas, N., 2019, December. LSTM based music generation. In 2019 International Conference on Machine Learning and Data Engineering (iCMLDE) (pp. 48- 53). IEEE

7. Kumar, S., Gudiseva, K., Iswarya, A., Rani, S., Prasad,K.M.V.V. and Sharma, Y.K., 2022, October. Automatic Music Generation System based on RNN Architecture. In 2022 2nd International Conference on Technological Advancements in Computational Sciences (ICTACS) (pp. 294-300). IEEE.

8. Liang, Tianmian, et al. "Research on Generating Xi'an Drum Music Based on Generative Adversarial Network." 2023 IEEE International Conference on Sensors, Electronics and Computer Engineering (ICSECE). IEEE, 2023.

9. Zhang, Haohang, Letian Xie, and Kaiyi Qi. “Implement music generation with gan: A systematic review.” 2021 International Conference on Computer Engineering and Apllication (ICCEA). IEEE, 2021.